Scripting Deployment of NIST Use Cases

Badi' Abdul-Wahid[1], Hyungro Lee[1], Gregor von Laszewski[1], and Geoffrey Fox[1,*]

[1]School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*]Corresponding authors: gcf@indiana.edu

January 20, 2017

This document summarizes the NIST Big Data Public Working Group (BDPWG) use-cases document: "Possible Big Data Use Cases Implementation using NBDRA". Additional projects originated from classes taught at Indiana University. The focus of our current work is the development of abstractions capturing the basic concepts we have seen and the development of a REST API implementing some of these ideas. We describe a few of the underlying components for allocating and managing virtual clusters. Additionally, a summary of the ecosystem (Open Source via GitHub) and Academic (via Indiana University classes) in which Big Data Software is used. There is a brief description of a few of the Use Cases, as well as a description of the major components. In summary: we have implemented two of the NIST Use Cases, with several others under development. Additionally, we are currently implementing DevOps support into Cloudmesh to facilitate scripting of Use Case deployment as started on an API to be exposed as a web service for deployments. An ecosystem analysis of projects on GitHub and classes show a preference for C++, Python, and Java as programming languages, as well as several popular packages as dependencies.

Keywords: Cloud, Big Data

https://github.com/cyberaide/nist-report

1. OVERVIEW

The goal of our work has been to try to understand some common themes among Big Data processing pipelines, as well as implement and use software allowing scripting to deploy, configure, and run such pipelines across a range of resource providers.

To this end, we have implemented the Fingerprint Matching and Face Detection use cases from the Big Data Working Group proposed Use Cases document and are in the development stage of several others: Twitter Analysis, Analytics for Healthcare Informatics, and Spatial Big Data Analytics. Additionally, we have investigated publicly available source code from GitHub as well as relevant student projects from classes taught at Indiana University to better understand the ecosystem of Big Data Analytics.

Finally, we are currently developing a software layer to expose as a REST API an interface that allows a kind of "Big Data as a Service" which takes advantage of the scripting capabilities currently under development.

Automating deployments involves many components:

Support infrastructure. A platform for allocating necessary compute, network, and storage resources. Examples include Amazon EC2 and Microsoft Azure, but may include physical machines as well as virtual ones.

Resource allocations. Groups of machines (or a virtual cluster) which can be configured to the desired state. These machines are provided by the support infrastructure.

Droops tools. By storing the desired state of the cluster in text file, these definitions can be iteratively improved. Examples include Ansible, Chef, and Puppet.

Domain knowledge. Important for having understanding configuring various components of the software stack to adequately support the task on hand. Additionally, domain knowledge supports the verification of development of the tools.

Pursuant to these components we are developing Cloudmesh Client, which interfaces with various support infrastructure to provide resource allocations. Current work focuses on creating abstractions to support an Application Programming Layer (API) which may be exposed using Representation State Transfer (REST) protocols. Key to this are the support of plugins, deployers, and compositions, which are described next.

Plugins are intended to encourage code reuse and sharing. Previously we were developing a single Ansible repository with the software modules such as Spark, HBase, Drill, etc. We have come to understand that an improved approach is to support a plugin-based architecture for deployment components. These components can therefore more easily be integrated from various sources.

Deployers are intended to allow the use of multiple DevOps tools. We had previously settled on the use of Ansible for deploying software components. While Ansible continues to be the primary tool, we are designing in support for other systems as well. This would allow users more familiar with Chef or NixOps for example to take advantage of the domain they know better.

Stack Compositions are intended to allow various components of a software stack, which depend upon layers lower in the stack, to be deployed together. A crucial component of composition is the ability to link properties of the stack, which may depend on values not known until deployment.

2. FOCUS OF CURRENT WORK

We are currently working on developing a an API for managing the infrastructure and deploying software stacks to the cluster. This API will be exposed as a REST service. A preliminary set of components that has been developed as part of the API has been used by students in classes in the Fall of 2016 and expect even more to use these components in the Spring 2017 class [1].

3. ARCHITECTURE

To address the complexity of the development of the different roles and scripts users must not only deliver them and present them to the data scientists, but must be prepared based on DevOps founded principals that deploys and test the scripts in a continuous or on an on demand fashion (in case significant resources are used to test the application). Hence in addition to the development of the scripts to deploy the software, we must also develop scripts that automatically execute the application on existing data sets to verify the functionality upon changes in the application, data, operating system, or infrastructure services such as hypervisors and containers.

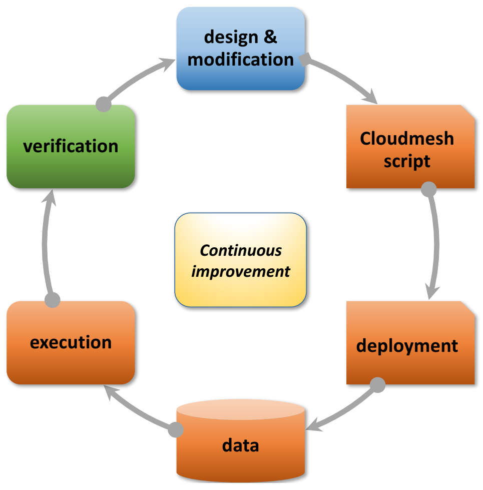


Fig. 1. Continuous improvement while using cloudmesh interactively.

Figure 1 depicts a high level overview of the cyclic workflow associated with a single application for a specific dataset. Starting with a Cloudmesh Script that allocates resources and then deploys the necessary software and desired dataset. The analytics component executed is then executed. Upon verification of the results and modification of components as needed the process repeats. This continues iteratively until the desired results are achieved. Tests at the end that verify certain results will be used to verify if the resulting run was successful. If not they are flagged and a notification is send to the maintaining team. New big data tests can be integrated into the set of benchmarks tested and the they can be evaluated based continuously or upon change.

This behavior is augmented in Figure 2 with the integration of the data repositories that are used to manage and maintain the scripts, the playbooks and the data. While the scripts are simple cloudmesh scripts to drive a particular application, we

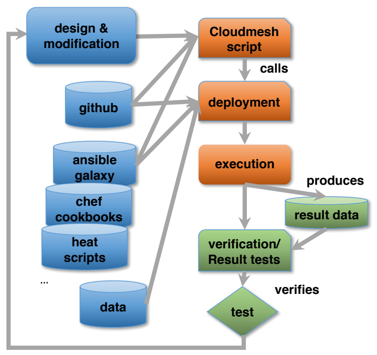


Fig. 2. Interaction of the continuous improvement steps with various databases while using Ansible deployment scripts.

envision that most roles are published as Ansible roles to be integrated in application specific playbooks. Playbooks exists for the phases of the execution

Phase 0: start of the integrated application verification
Phase 1: setup of the infrastructure
Phase 2: deployment of the platform and application
Phase 3: running a test on a specific data set.
Phase 4: verification of the result

4. SOFTWARE STACKS

We will be describing multiple software components in later sections. To put them into better context we provide a brief overview here.

Support infrastructure. Systems such as OpenStack, Amazon EC2, Amazon Azure, Google Compute Engine, Chameleon Cloud, Jetstream, San Diego Supercomputer Cluster's Comet. These are platforms that support allocating resources for computation, storage, and network.

DevOps. Tools such as Ansible, Puppet, Chef, and others listed in Table 1 provide the ability to install software on and configure a virtual cluster allocated by the support infrastructure.

Cloudmesh. A client program which interfaces with multiple support infrastructure providers, allowing allocation of virtual clusters across multiple providers. This greatly aids the goal of scripting deployments. Additionally, pursuant of the same goal, Cloudmesh is beginning to use DevOps tools to configure the allocated resource.

Big Data Stack. A collection of curated Ansible playbooks for deploying common Big Data software such as Hadoop, Spark, or HBase. It is developed the Cloudmesh group and is used to support the DevOps portion of the Cloudmesh codebase.

5. PROVISIONING MACHINES

The first step in a data processing pipelines is the provisioning/allocation of compute resources. These can be physical machines or, increasingly common and accessible, virtual machines. There are several major technologies that support this infrastructure as a service paradigm. Some notable ones include Amazon EC2, Microsoft Azure, Google Compute Engine, and Chameleon Cloud.

Part of our work has been in developing the cloudmesh client, which is a lightweight client interface of accessing heterogeneous clouds, clusters, and work- stations right from the users computer. The user can man- age her own set of resources she would like to utilize. Thus the user has the freedom to customize their cyber-infrastructure they use. Cloudmesh client includes an API, a command-line client, and a command-line shell. It strives to abstract backends to databases that are used to manage the workflow utilizing the different infrastructure and also the services. Switching for example to stage virtual machines from OpenStack clouds to amazon is as simple as specifying the name of the cloud. Moreover, cloudmesh client can be installed on Linux, MacOSX, and in future Windows. Currently cloudmesh supports backends to SLURM, SSH, OpenStack, Amazon EC2, and Azure. Using Cloudmesh, users can migrate across infrastructure service providers relatively seamlessly.

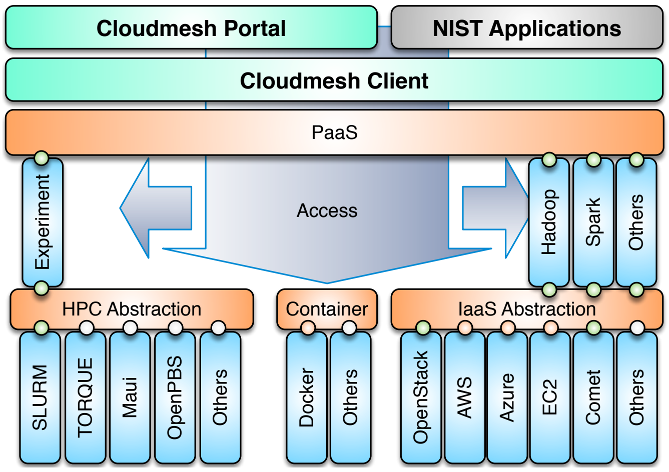


Fig. 3. Cloudmesh layered architecture.

Figure 3 describes the architecture of Cloudmesh Client.

6. DEPLOYMENT AND CONFIGURATION

The second step of the pipelines consists of deploying the desired software onto newly allocated resources. At this point, nodes may be available, but are in their initial state, which needs to be updated by installing and configuring software.

DevOps tools aim to support this component. One significant benefit is the conceptualization of /em code-as-infrastructure, that is the definitions of the desired state of the clusters are defined in text files that can be checked into a version control system and tested.

A. DevOps Tools

The following table provides a short overview of several notable DevOps tools. These tools allows the state of a cluster to be defined at various levels. Some (such as Ansible, Salt Stack) primarily operate at the level of software deployment and configuration of a preexisting cluster, while others (i.e. Juju, NixOps) additionally provide support for allocating the compute resources. Another factor is the interaction with the cluster via pushing the desired state to the nodes, or having the nodes run an agent that pulls the state definition and then evaluates it locally. Finally, the desired state is either declaratively defined, where the tools determines the steps required to achieve the state, or imperatively, where the user/developer is responsible.

Several of these tools may be used to showcase different capabilities:

Ansible is a YAML-based declarative descriptions of the desired state of the system. The description is used to generate a Python program that is pushed over SSH to each node, where it is then run to modify the system.

Chef is a Ruby Domain Specific Language for defining sets of actions (or /em recipes). These recipes are pulled onto the managed nodes from a configuration server.

Juju focuses on connecting services together, using arbitrary tools to bring a system to the desired state.

NixOps+NixOS provide resource allocation (EC2, GCE, Azure) for NixOS nodes as well as DevOps deployment. Once started, the nodes are brought to the desired state by interpreting a configuration file which declaratively defines the desired state of the node.

B. Concepts

One of our design principles is to allow the user to use whichever technologies they are most familiar with. Therefore we do not wish to constrain them into using a specific DevOps technology such as Ansible or Chef. Additionally, there are many deployment definitions for various technologies publicly available for these deployment software, as well as private definitions housed by whichever company developed them.

Cluster A cluster is a collection of virtual machines that can be references as a group. Methods acting on the virtual cluster are restricted to adding and removing instances. This layer is responsible for allocating and deallocating resources (i.e. starting and stopping the instances).

Stack. A stack defines how a particular technology is to be deployed. This uses deployer plugins to support using Ansible Roles, Chef Recipes, NixOS configurations, etc. The core concept is that a Stack represents a single technology, such as Hadoop, or Spark, or HBase. One of the desired properties of Stack evaluation is idempotence. Evaluating a given Stack on a node multiple times should have the exact same effect on the state of the system as evaluating it once.

Composition. A Composition is a group of Stacks that as a whole represent the deployment of a group of technologies. For instance Figure 4 shows that in order to provide Fingerprint Matching as a service, one would need a Web stack, a Fingerprint stack, and the Hadoop stack.
Figure 4. Fingerprint Stack composed with Hadoop and Web stacks to provide Fingerprinting-as-a-Service. Note that Stacks can also be Compositions
One crucial concept is that Stacks and Compositions are mutually recursive: a Composition is composed of at least one Stack, and each stack itself be a composition.

Link. A Link allows services to introspect properties of the virtual cluster and other Stacks at the time of deployment. For example, during the deployment of Apache Hadoop, the Hadoop Stack needs to determine the address of the nodes running the Zookeeper service, as well as the port on which Zookeeper is exposed. Currently this information is maintained by hand, which causes deployments to be very sensitive to how dependent compositions are configured. The goal for Linking is to propagate this information throughout the Stack Composition dependency graph as needed.

Deployment/Evaluation. A Deployment defines DevOps application-specific interface to evaluate Compositions. Figure 5 shows how the sample Fingerprint service stack

Table 1. List of notable DevOps tools.

Tool	Developer	License	Method	Approach
Ansible	Ansible Inc.	GPLv3	Push	Declarative
Chef	Chef	Apache v2	Pull	Imperative
Puppet	Puppet	GPL, Apache	Pull	Declarative
CFEngine	CFEngine AS	GPLv3	Pull	Declarative
SaltStack	Thomas Hatch	Apache v2	Push+Pull	Declarative+Imperative
Sup	Pressly, Inc	MIT	Push	Imperative
Fabric	Jeffrey Forcier	Open Source	Push	Imperative
Otter	Inedo	Proprietary	Push	Declarative+Imperative
Juju	Canonical	Affero GPL, LGPL	Push	Declarative
NixOps	NixOS Community	GPLv3	Push	Declarative

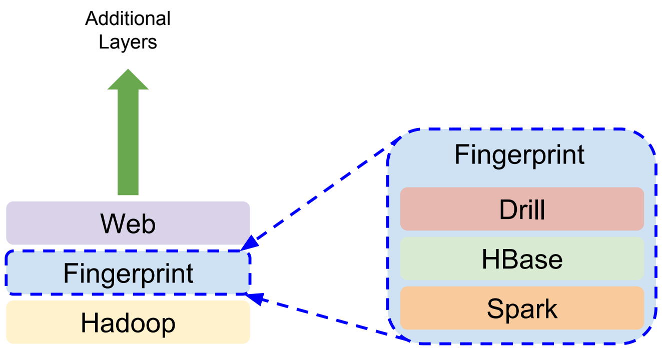


Fig. 4. Multiple stacks may be composed. Here, the *Fingerprint* stack is a composition of the Apache *Drill*, *HBase*, and *Spark* stacks. The *Fingerprint* stack is then included as a layer withing another composition built upon *Hadoop* and extended with the *Web* layers. The *Web* layer itself may be composed of other layers.

shown in Figure 4 may be evaluated. First the Daemon Watcher stack is deployed, as it is required by Hadoop. Although Java Stack is required for Hadoop, Spark, HBase, and Drill, it should only be evaluated once per node. The Fingerprint and Web stack may potentially be evaluated in parallel. The semantics of the parallel evaluation is not well understood at the moment though.
Figure 5. Evaluation of a Stack shown as a dependency graph.

C. Design Principles

Client based. As a client, the cloudmesh program requires no special privileges to run. This adheres to the principle of least privilege in an attempt to minimize security issues. The program runs locally end exits completely once the user terminates it.

Command Shell and Command Line Interface. The enables scripting with cloudmesh as a way of automating virtual cluster deployment and tear down.

Plugin Architecture. The purpose of the plugins is to allow users to use the deployment tools they are most familiar with, without constraining them to a specific technology. Additionally, a desired outcome of this design is increased code sharing and reuse.

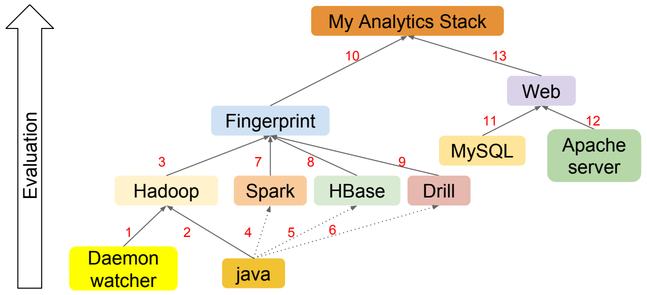


Fig. 5. Evaluation of a stack. The *My Analytics Stack* is composed of the *Fingerprint* and *Web* layers. These in turn are themselves compositions. Numbers indicate order of evaluation. While *spark*, *hbase*, and *drill* depend on *java*, re-evaluation (*4 - 6*) is not problematic due to the idempotency property.

REST API. The goal here is that the client may be used to support service-based deployments, in scenarios that warrant it.

Layered Architecture. The layered architecture, as shown in Figure 3, is intended to facilitate development and integration of different components that are not currently supported. For example, adding Docker support exposes a Docker layer that may be used to launch containers in a similar fashion to OpenStack virtual machines.

Management Framework. The supports management of allocated resources, allowing starting, stopping, restarting, reinitializing, etc.

Database Agnostic. The state of various components is stored in a database. Currently this is an SQLite database, but access to the database all passes through a dedicated module. The hope is that, if needed, the database backend may be parameterized to support others such as MariaDB, PostgreSQL, MongoDB, etc.

D. REST API Development

As described elsewhere, one of our goals is to provide a Representational State Transfer (REST) API for cloudmesh. This would enable web services to more easily use Cloudmesh to provide users with the ability to use and manage virtual clusters. REST is a well understood and widely used approach for designing and developing service-based APIs. Additionally the semantics of HTTP methods in a REST context are defined.

Table 2 shows an initial draft of the Cloudmesh REST API. These show resources and how the HTTP methods may be used to manage virtual clusters, stack compositions and deployments, and HPC experiments.

7. ECOSYSTEM ANALYSIS

We believe that big data ecosystem consists of various software, applications and datasets on different platforms. To understand current activities on big data projects and provide recommended software components (roles) in big data, we conduct analysis on big data projects 1) from community (i.e. github), 2) and academia (i.e. Indiana University) regarding to the following entities:

- development language preference
- library/package/tool dependencies
- sectors of public dataset source

This effort will result in building recommended software components (roles) which supports most of functionalities in a given big data applications.

A. Analysis on Big Data Projects from Community

Github.com has been used to provide version control and manage source code development along with diverse collaborators across countries. The popularity of github as a collaboration tool has been significantly increased and 4,995,050 repositories exist as of 12/27/2016 with 20-30 thousands daily added repositories. To understand trends on big data software development from community, we conducted a survey of github repositories regarding to big data applications and tools. Every github repository has a description of a project and we searched them using topic keywords. For example, we collected github repositories for Face Detection with search keywords; face detection, face recognition, human detection, and person detection to conduct a survey on a series of questions regarding to 1) A development language distribution, 2) dependency of libraries and packages, and 3) sectors of public dataset. Actual source code of public github repositories are evaluated with the survey query data available on https://github.com/lee212/bd_stats_from_github. There are six topics of NIST Collection used in this analysis where N1: Fingerprint Matching, N2: Face Detection, N3: Twitter Analysis, N4: Data Warehousing, N5: Geographic Information Systems, and N6: Healthcare Data. In addition, a list of recommended components (roles) is created based on the survey results.

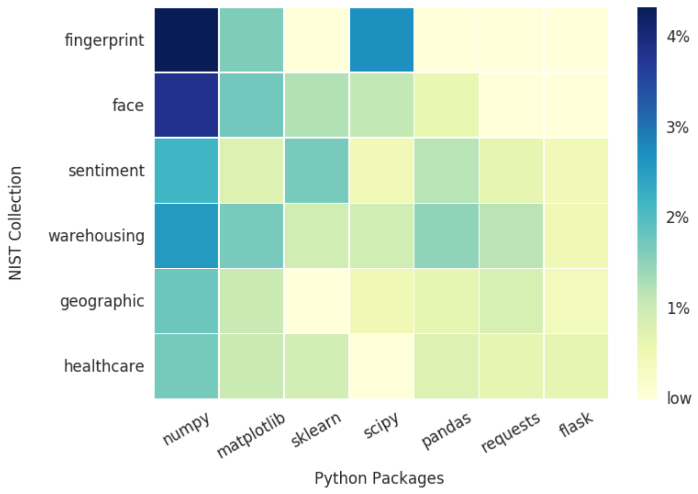
A.1. Language Preference

The repository statistics indicate that C++, Python and Java are most common languages among the NIST collection (Table 3), although Matlab is dominant in the fingerprint.

A.2. Package/Library/Tool Dependencies

We also notice that scientific python packages are commonly used to enable numerical computation, data analysis and visualization for these big data applications (Figure 6), whereas there are dependent packages for each project (Table 4). Tweepy, twitter API, is used in the twitter live analysis projects with NLTK, the natural language processing toolkit to complete sentiment analysis with streaming data, tweets. Similarly, GIS projects use particular libraries for spatial analysis such as geopy and shapely. New trends in software development packages and libraries are observed, for example, deep learning python packages e.g. caffe or theano have added recently to github repositories. Statistics

Fig. 6. Python Packages used in NIST Collection (collected from Github)



(tables5) show that popular github repository examples related to the six NIST projects started in 2016. Each github project has different language preferences with various libraries and packages however similar trends are observed, for example, deep learning software such as Keras, Theano, mxnet and Caffe are adopted among multiple projects.

Additional packages (table 4) are not required in a default set of big data ecosystem but it is necessary to indicate the dependency with a particular application.

A.3. Datasets

Finding relevant datasets for particular applications is another challenge for the big data ecosystem because of its difficulty of collecting data from multiple sources (Kim, Trimi, and Chung, 2014), complexity and diversity (Hashem et al., 2015). Community contributed lists of public datasets (Cohen and Lo,2014) provide structured information with a specific location to access data and a category to describe itself. We intend to generate linked json data for datasets and applications in big data ecosystem based on these lists because it connects scattered data and software in an organized way. Table 6 shows the data source from different sectors, academia(.edu or .ac), government(.gov), organization(.org), industry(.com or .net), and international(country code in suffix), among the seven topics of the lists. Entire topics are available online: https://github.com/lee212/bd_datasets.

B. Analysis on Big Data Projects from Academia

At Indiana University, Big Data Analytics and Applications (BDAA) and Big Data Open Source Software Projects (BDOSSP) have been offered in 2015 and 2016 accordingly. During these semesters, students have asked to complete a course project with big data software, tools and datasets. The choice of languages, software, and dataset are surveyed in this section.

B.1. Language Preference

Table 7 shows the language distribution for Big Data Classes from Indiana University.

Table 2. Selected Service Description. Items prefixed with a colon (:) indicate parameters e.g. :id, :?.

Resource	REST Method	Description
Virtual Cluster: /cluster		
/	GET	List available clusters
/	POST	Launch a cluster on the provider
/	DELETE	Delete all available clusters
/:id	DELETE	Delete and destroy a cluster
/:id	GET	View the status of a cluster (nodes, node type, etc)
/:id/properties/:property	GET, PUT	Get/set a property (provider, name, description) of the cluster
/:id/inventory/:format	GET	Obtain an inventory of the cluster
Stack Composition: /stack		
/	GET	List available compositions
/	POST	Create a new composition
/:id	GET	Show information about the composition
/:id	DELETE	Delete a composition
/:id/name	GET, PUT	Get/set the name of the composition
/:id/add?deployer=:?&source=:?	POST	Add a layer to the composition
/:id/layers	GET	List layers of the composition
/:id/layers/:id	DELETE	Delete the layer of the composition
Stack Deployment: /stack		
/	GET	List the available stacks with description
/:id/deployments/:cluster	POST	Deploy a stack onto a cluster
/:id/status	GET	Current status
/:id/deployments/:cluster	GET	Current status on given cluster
Batch Experiments: /hpc		
/	GET	List all jobs started with the run command
/:id	DELETE	Deletes the experiment with the given id
/run?script=:?&cluster=:?	POST	Submits an experiment to the named cluster
/:id/status	GET	Returns the status of the job started with the run command

B.2. Package/Library/Tool Dependencies

We had 37 final projects from Big Data Open Source Project Spring 2016 and Table 9 shows that packages/tools/libraries used in the class projects. Apache Hadoop is mostly used in conducting data analysis with a database support from HBase, Hive and HDFS. Python was the most preferred language in the course projects which resulted in high use of Spark in data processing with the python library, pyspark. One another observation is that Ansible, software deployment tool, had offered as a method of project deployment to ensure reproduceability.

B.3. Datasets

There were 49 class project in Big Data Analytics and applications Fall 2015, and use of 27 dataset are observed. Public dataset from industry was mainly used (44%) due to the interest on analytics from kaggle and twitter and availability e.g. amazon reviews and yelp reviews.

8. COMPLETED APPLICATIONS

A. Fingerprint Matching

Implementation repository
https://github.com/cloudmesh/example-project-nist-fingerprint-matching

A.1. Description

Fingerprint recognition refers to the automated method for verifying a match between two fingerprints and that is used to identify individuals and verify their identity. Fingerprints (Figure 7) are the most widely used form of biometric used to identify individuals.

The automated fingerprint matching generally required the detection of different fingerprint features (aggregate characteristics of ridges, and minutia points) and then the use of fingerprint matching algorithm, which can do both one-to- one and one-to-many matching operations. Based on the number of matches a proximity score (distance or similarity) can be calculated.

The goal for this project is: given a set of probe and gallery images, compare the probe images to the gallery images, and report the matching scores. The dataset used comprises 54,000 images along with their metadata. Provided tools include MINDTCT

Table 3. Language Distribution of Topics related to those in the NIST collection on Github

Topic	C++	Python	Java	Matlab	JS	C#	C	R	Ruby	Scala	Count*
Fingerprint	15%	11%	13%	20%	3%	16%	8%	0%	1%	5%	43
Face	26%	21%	12%	9%	7%	5%	2%	2%	1%	.02%	538
Twitter	2%	35%	15%	.6%	9%	2%	1%	10%	3%	1%	1429
Warehousing	3%	27%	18%	2%	10%	3%	1%	10%	4%	1%	3435
Geographic	5%	15%	27%	4%	15%	3%	5%	7%	3%	16%	6487
Healthcare	2%	13%	19%	2%	14%	5%	1%	10%	6%	2%	132

* Count: average number of github.com repositories.

Table 4. Additional Python packages found in NIST Collection

Python Package	Description	Fingerprint	Face	Twitter	Warehousing	Geographic	Healthcare
cv2	OpenCV	✓	✓				
skimage	Image Processing		✓				
PIL	Python Imaging		✓				
caffe	Deep Learning		✓				
nltk	Natural Language Toolkit			✓			
tweepy	Twitter			✓			
Beautiful Soup	Screen scraping			✓	✓		
gensim	Topic Modelling			✓	✓		
geopy	Geocoding library					✓	
shapely	Geometric Analysis					✓	
django	Web framework				✓		✓

and BOZORTH3, which are part of the NIST Biometric Image Software (NBIS) release. These two tools form the core of the application: MINDTCT preprocesses the images to identify minutae which is used by BOZORTH3 to compute a match.

The implemented solution uses stack of HDFS, YARN, Apache Spark, Apache HBase, and Apache drill. A Hadoop cluster is deployed and YARN used to schedule Spark jobs that load the images into HBase, process the images, and compute the matches. Apache Drill, with the HBase plugin, can then be used to generate reports.

A.2. Software Stack

- Big Data software packages:
  - Apache Hadoop (YARN)
  - Apache Spark
  - Apache HBase
  - Apache Drill
- Datasets:
  - NIST Special Database 14 - Mated Fingerprint Card Pairs 2.
- Domain Specific code (NBIS):
  - MINDTCT
  - BOZORTH3
- Other tools and technologies:
  - Scala

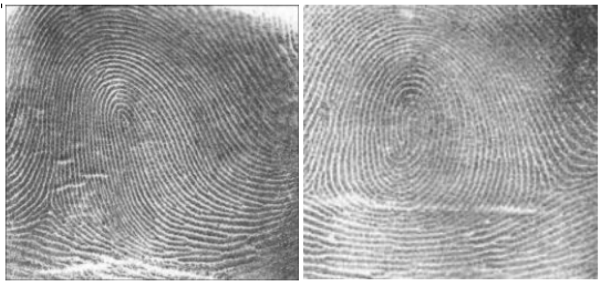


Fig. 7. Example Fingerprints.

A.3. Deployment Approach

The resource allocation can be done using Cloudmesh Client. Next, Cloudmesh Big Data Stack is used to deploy the Big Data software packages. Finally, some Ansible playbooks deploy and compile a Scala program that integrates the Big Data infrastructure with running the domain specific code.

B. Face Detection

Implementation repository:
https://github.com/futuresystems/pedestrian-and-face-detection

B.1. Description

Human detection and face detection have been studied during the last several years and models for them have improved along with Histograms of Oriented Gradients (HOG) for Human Detection [2]. OpenCV is a Computer Vision library including the SVM classifier and the HOG object detector for pedestrian detection and INRIA Person Dataset [3] is one of popular samples for both training and testing purposes. This example shows how to deploy the NIST Human and Face Detection with INRIA Person Dataset to the cluster where we deployed Apache Spark on Mesos to train and apply detection models from OpenCV using Python API. OpenCV Python code runs with Spark Map function to perform distributed job processing on the Mesos scheduler.

B.2. Software Stack

- Big Data software packages:
  - Apache Spark
  - Apache Mesos
  - Apache Zookeeper
  - OpenCV (with Python)

Table 5. Example Projects Recently Created Regarding to Face Detection

Title	Description	Language	Start Date	Popularity	Dependency
OpenFace	an open source facial behavior analysis toolkit	c++	March, 2016	725 (305)	OpenCV, dlib, boost, TBB
Picasso face detection transformation	An Android image transformation library providing cropping above Face Detection (Face Centering) for Picasso	Java	July, 2016	528(56)	Square Picasso
MTCNN face detection alignment	Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Neural Networks	Matlab	September, 2016	226(162)	Caffe, Pdollar toolbox
facematch	Facebook Face Recognition wrapper	JavaScript	January, 2016	132 (41)	fbgraph, request, body-parser, express
mxnet mtcnn face detection	MTCNN face detection	Python	October, 2016	99 (47)	OpenCV, mxnet

Table 6. Public Dataset of Academia, Government, Organization, Industry and International from Community

Category	Academia	Government	Organization	Industry	International	Total
GIS	1	3	5	9	5	23
Healthcare	0	6	3	1	1	11
Image Processing	11	0	4	2	5	18
Natural Language	7	0	8	7	6	26
Social Networks	8	0	7	5	5	24
Climate/Weather	2	6	3	2	4	16
Energy	2	2	5	1	5	15

Table 7. Language Distribution for Big Data Classes from Indiana University

Class	Java	Python	R	C#	Projects Count
Fall '15	6	29	10	1	49
Spring '16	11	16	1	0	37
Fall '16	1	73	3	0	77

- Datasets:
  - INRIA Person Dataset

B.3. Deployment Approach

Mesos role is installed with two Ansible inventory groups; masters and slaves where Mesos-master runs on the masters group and Mesos-slave runs on the slaves group. Apache Zookeeper is included in the Mesos role and Mesos slaves find an elected Mesos leader from the zookeeper. Spark, as a data processing layer, provides two options for distributed job processing, batch job processing via a cluster mode and real-time processing via a client mode. The Mesos dispatcher runs on a masters group to accept a batch job submission and Spark interactive shell, which is the client mode, provides real-time processing on any node in the cluster. Either way, Spark is installed after a scheduler layer i.e. Mesos to identify a master host for a job submission.

Table 8. List of Top 15 Tools/Libraries/Packages used in Big Data Class Fall 2015

Package	Type	Use	Language	Count
Numpy	library	data management	Python	13
Pandas	library	data management	Python	10
Matplotlib	library	visualization	Python	8
Scipy	library	data management	Python	6
Mongodb	database	NoSQL	C++	10
Hadoop	framework	parallel processing	Java	6
Nltk	library	NLP	Python	5
ggplot2	library	visualization	R	4
scikit-learn	library	machine learning	Python	7
IPython	Tool	Web Interface & Notebook	Python	4
Tableau	Tool	visualization	C++	4
rstudio	Tool	development IDE	R	3
seaborn	library	visualization	Python	3
Spark	framework	in-memory processing	Scala	3
Randomforest	library	classification	R	2
Hbase	database	NoSQL	Java	2
Folium	library	visualization	Python	2
MPI	framework	parallel processing	C, C++, Fortran	2
Pig	language	data processing	Java	2

Count: a number of class projects in a given tool/library/package

Installation of OpenCV, INRIA Person Dataset and Human and Face Detection Python applications are followed.

9. APPLICATIONS UNDER DEVELOPMENT

A. Twitter Analysis

A.1. Description

Social messages generated by Twitter have been used with various applications such as opinion mining, sentiment analysis (Pak and Paroubek, 2010), stock market prediction (Bollen, Mao, and Zeng, 2011), and public opinion polling (Cody et al., 2016) with the support of natural language tool-kits e.g. nltk (Bird, 2006), coreNLP (Manning et al., 2014) and deep learning systems (Kim, 2014). Services for streaming data processing are important in this category. Apache Storm is widely used with the example of twitter sentiment analysis, and Twitter Heron, Google Millwheel, LinkedIn Samza, and Facebook Puma, Swift,

Table 9. List of Top 15 Tools/Libraries/Packages used in Big Data Class Spring 2016

Package	Type	Use	Language	Count*
Hadoop	framework	parallel processing	Java	31
Ansible	tool	deployment	Python	26
Spark	framework	in-memory processing	Scala	14
HBase	database	NoSQL	Java	12
Pig	language	data abstraction	Java	11
Hive	database	SQL	Java	7
MongoDB	database	NoSQL	C++	7
Mahout	library	machine learning, data mining	Java	4
MLLib	library	machine learning	Java	4
OpenCV	library	computer vision	C++	3
Zookeeper	framework	directory service	Java	3
Tableau	tool	visualization	C++	3
D3.js	tool	visualization	Javascript	2
MySQL	database	SQL	C++	2
HDFS	database	distributed filesystem	Java	2

* Count: a number of class projects in a given tool/library/package

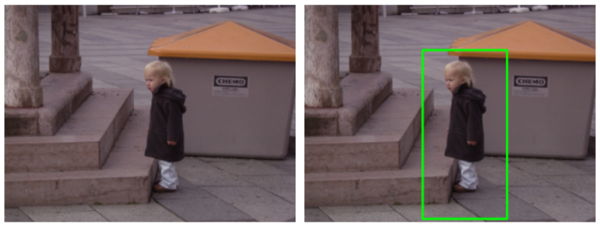


Fig. 8. Human Detection, original image (left), processed image (right)

and Stylus are available as well [4].

A.2. Software Stack

Big Data software stack:

- Apache Hadoop
- Apache Lucene
- Twitter Heron
- Apache Storm
- Apache Flume
- Natural Language Toolkit (Python)
- gensim NLP library

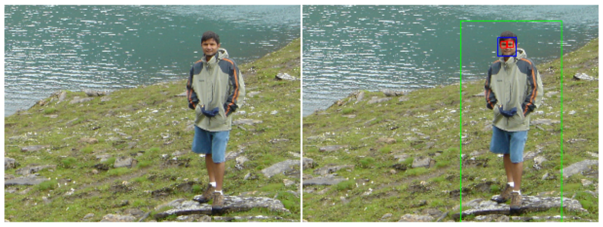


Fig. 9. Face and Eye Recognition with Human Detection (face: blue box, eye: red box, human: green box)

Table 10. List of Top 15 Tools/Libraries/Packages used in Big Data Class Fall 2016

Package	Type	Use	Language	Count*
Matplotlib	library	visualization	python	88
Pandas	library	data management	python	79
Numpy	library	data management	python	65
Scipy	library	data management	python	24
Requests	library	http tool	Python	22
xlrd	library	MS Excel	Python	10
pillow	library	imaging library	Python	9
scikit-learn	library	machine learning	Python	9
seaborn	library	visualization	Python	8
nltk	library	language processing	Python	6
geopy	library	geospatial tool	Python	5
pyOpenSSL	library	OpenSSL	Python	5
patsy	library	statistics	Python	5
bokeh	library	visualization	Python	4
ploty	library	visualization	Python	4

* Count: a number of class projects in a given tool/library/package

Table 11. Dataset Sectors of academia, government, organization, industry and international from Big Data Classes at Indiana University

Class	Academia	Government	Organization	Industry	International	Total
Fall ’15	7	3	5	12	0	27
Spring ’16	6	6	8	10	1	30

B. Analytics for Healthcare Data and Informatics

B.1. Description

Several attempts have been made to apply Big Data framework and analytics in health care with various use cases. Medical image processing, signal analytics and genome wide analysis are addressed to provide efficient diagnostic tools and reduce healthcare costs (Belle et al., 2015) with big data software such as Hadoop, GPUs, and MongoDB. Open source big data ecosystem in healthcare is introduced [5] with examples and challenges to satisfy big data characteristics; volume, velocity, and variety [6]. Cloud computing framework in healthcare for security is also discussed with concerns about privacy [7].

B.2. Software Stack

Big Data software stack:

- Apache Hadoop
- Apache Spark/mllib
- Apache Mahout
- Apache Lucene/Solr
- Theano deep learning library

C. Spatial Big Data/Statistics/Geographic Information Systems

C.1. Description

The broad use of geographic information system (GIS) has been increased over commercial and scientific communities with the support of computing resources and data storage. For example, Hadoop-GIS citeaji2013hadoop, a high performance spatial data warehousing system with Apache Hive and Hadoop, offers spatial query processing in parallel with MapReduce, and HadoopViz [8], a MapReduce framework for visualizing big spatial data, supports various visualization types of data from satellite data to countries borders.

C.2. Software Stack

- Apache Hadoop
- Apache Spark/mllib
- GIS-tools
- Apache Mahout
- GDAL - Geospatial Data Abstraction Library
- S2 Geometry Library
- geopy geocoding python library

D. Data Warehousing and Mining

D.1. Description

Researches in data warehousing, data mining and OLAP have investigated current challenges and future directions over big data software and applications [9] due to the rapid increase of data size and complexity of data models. Apache Hive, a warehousing solution over a hadoop [10], has introduced to deal with large volume of data processing with the other research studies [11, 12] and NoSQL platforms [13] have discussed with data warehouse ETL pipeline [14].

D.2. Software Stack

Big Data software stack:

- Apache Hadoop
- Apache Spark/mllib
- MongoDB
- Hive
- Pig
- Apache Mahout
- Apache Lucene/Solr
- MLlib
- Google BigQuery

10. APPLICATION STATUS

A. Deployment Approach

The resource allocation can be done using Cloudmesh Client. Next, Cloudmesh Big Data Stack is used to deploy the Big Data software packages. Finally, some Ansible playbooks deploy and compile additional packages that integrates the Big Data infrastructure with running the domain specific code.

B. Development Status

Fingerprint Matching: implementation completed complete reporting in desired NIST WG report format.

Facetetection: implementation completed, complete reporting in desired NIST WG report format.

Twitter analysis: prototyped Cloudmesh integration of DevOps approach, virtual cluster command implemented, Ansible plugin in development, Hadoop deployment prototyped and tested with students, application implementation under development. Multiple prototypes submitted by students

Analytics for healthcare data and Informatics: prototyped Cloudmesh integration of DevOps approach, virtual cluster command implemented, Ansible plugin in development, Hadoop deployment prototyped and tested with students, application implementation under development. A prototype submitted by students

Spacial big data / statistics / geographic information systems: prototyped Cloudmesh integration of DevOps approach, virtual cluster command implemented, Ansible plugin in development, Hadoop deployment prototyped and tested with students, application implementation under development. Prototype for earthquake data analysis propagating through seismic sensors.

Data warehousing and mining: prototyped Cloudmesh integration of DevOps approach, virtual cluster command implemented, Ansible plugin in development, Hadoop deployment prototyped and tested with students, application implementation under development. An prototype application to analyze Uber data is under review.

11. COMPONENTS (ROLES)

Based on the analysis from community and academia, we observed that there are crucial software, dataset and analytics in big data ecosystem. We, therefore, offer deployable first-class roles which enable major functionalities on big data processing and analysis. The software deployment is accommodated with Cloudmesh, Ansible, Chef, Puppet, Salt, OpenStack Heat, Microsoft Azure Template, and Amazon Cloudformation.

- Framework
  - Hadoop
  - Mesos
- Processing
  - Spark
  - Storm
- Language
  - Pig
  - R
- Database
  - HBase
  - Hive
  - MongoDB
  - MySQL
- Library
  - Mahout
  - nltk
  - MLlib
  - Lucene/Solr
  - OpenCV
- Tools
  - Ganglia
  - Nagios
  - Zookeeper

Table 12) compares dataset Sectors of academia, government, organization, industry and international from Big Data Classes at Indiana University.

ACKNOWLEDGMENTS

This work was in part supported by the NIST project *NIST-70NANB15H247*.

Table 12. Dataset Sectors of academia, government, organization, industry and international from Big Data Classes at Indiana University

Role Name	Description	Type	Requirement (Installation)	Dependencies	Distributed	Example
Spark	In-memory data processing application	processing	JDK 7+ Maven 3.3.9	Hadoop (optional)	Cluster Manager/Executor (Worker node)	
Hadoop	Map/Reduce distributed processing framework	framework	JDK	Resource Manager/Node Manager		WordCount
Storm	a real time fault-tolerant and distributed stream data processing system	processing	OpenJDK	ZooKeeper	Master/Worker	
Zookeeper	Synchronization service for distributed processes	tool	JDK 6+		Ensemble (3 servers minimum)	
HBase	NoSQL database for real-time processing	database	JDK	Zookeeper	Master / RegionServer	
Twitter REST APIs	Reading Twitter data	library		OAuth		
D3, tableau	Javascript visualization library	visualization				
Nltk	Natural Language Toolkit	library	Python2.7 or 3.2+	Numpy (optional)		
AlchemyAPI	Collecting Tweets	library	Mongodb, R (ggplot2)			
OpenCV	Computer Vision Libraries	library				
Mahout	Machine learning applications	library	JDK Maven	hadoop-client	Via Hadoop, Spark, H20 or Flink	Naive Bayes (Classification) K-Means (Clustering) Recommender
Lucene/Solr	Search engine framework	library	JRE 1.8+		SolrCloud	
MLlib	Machine Learning Library from Spark	library		Spark		Logistic regression (Classification) K-means (Clustering)
MongoDB	Document-oriented database	application			mongos/shard	
Hive	Database SQL query interface to apache hadoop	application	JDK Hadoop	Hadoop		
Pig	High level scripting language for Hadoop	Application	Hadoop JDK Ant (optional) JUnit (optional)			
RethinkDB	NoSQL, distributed document-oriented (JSON) database	application			sharding, replication	
TextBlob	Natural language processing (NLP) Python library	Analytics	NLTK corpora			
Pattern	Web mining python library	Analytics	Numpy (for LSA; Latent semantic analysis)	Python 2.5+ only, no support on 3		

REFERENCES

[1] Gregor von Laszewski and Badi Abdul-Wahid, “Big Data Classes,” Web Page, Indiana University, Jan. 2017. [Online]. Available: https://cloudmesh.github.io/classes/

[2] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05), vol. 1. IEEE, 2005, pp. 886–893.

[3] ——, “Inria person dataset,” 2005.

[4] G. J. Chen, J. L. Wiener, S. Iyer, A. Jaiswal, R. Lei, N. Simha, W. Wang, K. Wilfong, T. Williamson, and S. Yilmaz, “Realtime data processing at facebook,” in Proceedings of the 2016 International Conference on Management of Data. ACM, 2016, pp. 1087–1098.

[5] W. Raghupathi and V. Raghupathi, “Big data analytics in healthcare: promise and potential,” Health Information Science and Systems, vol. 2, no. 1, p. 1, 2014.

[6] P. Zikopoulos, C. Eaton et al., Understanding big data: Analytics for enterprise class hadoop and streaming data. McGraw-Hill Osborne Media, 2011.

[7] V. Stantchev, R. Colomo-Palacios, and M. Niedermayer, “Cloud computing based systems for healthcare,” The Scientific World Journal, vol. 2014, 2014.

[8] A. Eldawy, M. Mokbel, and C. Jonathan, “Hadoopviz: A mapreduce framework for extensible visualization of big spatial data,” in IEEE Intl. Conf. on Data Engineering (ICDE), 2016.

[9] A. Cuzzocrea, L. Bellatreche, and I.-Y. Song, “Data warehousing and olap over big data: current challenges and future research directions,” in Proceedings of the sixteenth international workshop on Data warehousing and OLAP. ACM, 2013, pp. 67–70.

[10] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, “Hive: a warehousing solution over a map-reduce framework,” Proceedings of the VLDB Endowment, vol. 2, no. 2, pp. 1626–1629, 2009.

[11] S. Chen, “Cheetah: a high performance, custom data warehouse on top of mapreduce,” Proceedings of the VLDB Endowment, vol. 3, no. 1-2, pp. 1459–1468, 2010.

[12] Y. He, R. Lee, Y. Huai, Z. Shao, N. Jain, X. Zhang, and Z. Xu, “Rcfile: A fast and space-efficient data placement structure in mapreduce-based warehouse systems,” in 2011 IEEE 27th International Conference on Data Engineering. IEEE, 2011, pp. 1199–1208.

[13] M. Chevalier, M. El Malki, A. Kopliku, O. Teste, and R. Tournier, “Implementing multidimensional data warehouses into nosql,” in 17th International Conference on Enterprise Information Systems (ICEIS’15), Spain, 2015.

[14] K. Goodhope, J. Koshy, J. Kreps, N. Narkhede, R. Park, J. Rao, and V. Y. Ye, “Building linkedin’s real-time activity data pipeline.” IEEE Data Eng. Bull., vol. 35, no. 2, pp. 33–45, 2012.